Itterative Development

Project 1

I didn't forget to change the title this time

Alex Harper

Contents

1 Introduction

I spent a long time reading the book, it has many many pages to cover for this first project. While I was reading, I kept putting it in terms of an idea I had been wanting to do for a little while now; making a "Reddit Bot". The idea is that you have a piece of software from time to time do automated tasks interacting with the site reddit. This is slightly nebulas as there are many things you could get it to do. Really I wanted to make a fairly general wrapper around the website actions that I can make do more interesting things later.

For this paper, it is probably going to be a bit shorter than the 20 pages I normally aim for as the minimum. The report will be about what I did and my thoughts about things, while there will be two other documents beside this report. Since this class is supposed to be about project managment with an iterative style, I believe creating the apporpriate documents for each time section is critical for understanding.

2 Inception

"This section is to be short and determine only if the project should continue and what are the surface level constraints."

That is not a direct quote but in general sums up what the book says. As such, in the document for it, many things are simple and fairly general. While I had done some basic looking around before writing it, I did not know exactly how things were going to happen.

2.1 Use Cases

It differs from the examples in the book in this section because the program is supposed to run without any user interaction. This does essentailly mean there is no "user" in our use cases, but is still in the spirit of what should be described. As such the section is used to describe the functional parts of the program in general.

This section has what I believe to be the core ideas of what the bot needs to be able to do. There are 4 main parts with the focus on the program being more of a framework. While there will be a main action that it does, I am considering it more of a test use rather than the end goal. These main actions are high level and should be broken down into more specific things during elaboration.

A 5th case is listed as a substep because it is so important; Authenticating with the servers. The two cases for dealing directly with reddit will both directly depend on the same version of this case, and so I think it is appropriate to pull it out on it's own.

2.2 Supplementary Specs

These are the non-functional requirements. Mostly these guide the behavior of the program, but are not directly what the program does. At some point, code to address some aspects of it will need to be implemented such as limiting the rate the API gets used. For example, limiting the rate of requests to abide by the rules can be enforced with an internal mechanism.

2.3 Proof of Concept

To make sure that what I was wanting to do was even possible, two small programs were made. I honestly was not expecting to be able to use the Qt framework for this, but apparently it works well enough. The main question it answered was how easy would it be to customize the fields for the HTTP request. Besides a little funny buisness with first time authentication, it is simple and easy. The second program was to see how nice the JSON objects were that Qt provided. They work well enough that I don't feel like putting the effort into parsing things myself.

Honestly I forgot to save this code. I am so used to having these tiny pieces be completly disposable that I didn't move them from my temp folder before reboting. It is no real big deal for the real project because I learned what needed to already and could do better the second time around.

3 Iteration 1

Now we get into the guts of the program; flushing out the most important bits of the documenation and trying to get the basics of the program working.

During this part I was able to accomplish the basics of accessing reddit. The program currently can authenticate as a user, get information using the user credentials, and restricts the API usage per time period to prevent abuse. This is mostly what was planned to be done for this iteration with an extra bit of enforcing the rules.

3.1 Use Cases

I filled out the main points of two use cases as the code directly does these two. The cases show the different thoughts and steps that should be taken (in general form) to do the right thing.

I had to take a slightly different approach than the book due to being a single person. Normally the head people will be making the main documentation for different parts and everyone else will be getting code done. This means that a few people grok what is supposed to happen, put it on paper, and then eventually it gets implemented. This leads to the documentation leading ahead of the code that is implemented. As a single person, I have to grok it before I can program it, but I might as well program it as I try to grok it to make sure I am doing things right. This has lead to me having the documentation matching the amount of what I have programmed.

3.2 Domain Diagram

This encompases the whole of what interacting with the reddit website will be like (probably). The diagram shows the replationships of different types of responses we can get from the server. The distinctions were layed out in the reddit documentation and has each returned item is tagged with a field to let us know what is what.

For the next iteration, I will make a second domain diagram for the imgur website and possibly a third for the specific use of the robot that is being used as the test.

3.3 Sequence Diagram

There is only this single sequence diagram because all the others would look almost identical at this point. (make a session, pass to the worker object, ask object to get something, get the response) There will be more things that can get a sequence diagram for the next iteration as things will get more varied.

4 Where am I?

At this point, the basic work of connecting to reddit and getting responses for some basic aspects of the site is done.

- The responses from the website are easily parsed by the JSON objects provided by Qt.

- The requests to the servers is transparently forced to follow the rules of the site using the session object.
- The session object at a later point could be extended to support multiple different users.
- Conveniance objects are ready to be filled out for the common uses, such as getting links from a subreddit.

So for the next iteration, the short list of what I want to get done.

- Fill out the conveniance objects.
- Get a similar system setup for posting and parsing the imgur website. (likely will be a single object)
- Parse reddit messages and respond to them.

A redditapi.h

```
#ifndef REDDITAPI_H
#define REDDITAPI_H

#include <QObject>
#include <QDir>
#include <QString>
#include <QCoreApplication>
#include <stdlib.h>
#include <stdio.h>
#include <string>
#include <iostream>
#include <QElapsedTimer>
#include <QTimer>

#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkReply>
#include <QUrl>
#include <QAuthenticator>
#include <QNetworkCookieJar>

#include <QJsonDocument>
#include <QJsonArray>
#include <QJsonValue>
#include <QJsonObject>

#include "configFileParser/glopConfig.h"
/*t1*/ class Reddit_Comment;
/*t2*/ class Reddit_Account;
/*t3*/ class Reddit_Link;
/*t4*/ class Reddit_Message;
/*t5*/ class Reddit_Subreddit;
/*t6*/ class Reddit_Award;
/*Listing*/ //has another member "data" who will have another object children w

class RedditSession : public QObject{
    Q_OBJECT
    static const QString clientID;
```

```
39      static const QString UserAgent;
40      static const QString redirectUri;
41      QString API_Token; //permanent token to get a temperary token with
42      QString API_Session; //temperary session key
43      static QNetworkAccessManager *networkManager;
44
45      QElapsedTimer Session_Time_Used; // to compare to for knowing when to refresh
46      QElapsedTimer RequestLimit_TimeElapsed; //elapsed time since we were told any
47      int RequestLimit_RequestsLeft; // how many times the API can be used before u
48      int RequestLimit_TimeToReset; // time we needed to wait for the limit to be r
49
50      void commonRequestSetup(QNetworkRequest &req,QString URL);
51      void commonReplyWait(QNetworkReply *reply);
52      void waitForRequestLimitReset();
53  public slots:
54      // These are the only functions you should need to call
55      bool parseSettings(bool allow_example_firstRun = true); // returns false if u
56      void saveSettings();
57      QNetworkReply* makePOSTrequest(QString URL, QString postData);
58      QNetworkReply* makeGETrequest(QString URL);
59  public slots:
60      void firstTimeRun_example(); // prints things out and has the user do things
61      QString firstTimeRun_authorizeURL(QString wanted_scopes); //have the user vis
62      void firstTimeRun_redirectedURL(QString URL); // after the user says yes, the
63      QString scopesDescriptions();
64
65      void testFunction(); // for me to test things with and random junk in it
66
67      void refreshSession();
68      void sendAuthentication(QNetworkReply * reply, QAuthenticator * authenticator)
69
70  public:
71  };
72
73  #endif // REDDITAPI_H
```

B redditapi.cpp

```cpp
#include "redditapi.h"

const QString RedditSession::clientID = "f3dZ8gIetXY69Q";
const QString RedditSession::UserAgent = "pc:glop-bot:v0.0 (by /u/glop102)";
const QString RedditSession::redirectUri = "http://www.reddit.com";
//QString RedditSession::API_Token=""; //identifies that the user authorised our
//QString RedditSession::API_Session=""; //temperary session key
//QTimer *RedditSession::Session_Time_Used = new QTimer();
QNetworkAccessManager *RedditSession::networkManager = NULL;

bool RedditSession::parseSettings(bool allow_example_firstRun){
    GlopConfig::Settings set = GlopConfig::ParseFile((QDir::homePath()+"/.redditSe
    Session_Time_Used.start();
    RequestLimit_TimeElapsed.start();
    RequestLimit_RequestsLeft = 60; // mediocre default that will be fixed on the
    RequestLimit_TimeToReset = 10;

    API_Token = QString::fromStdString( set.getValueAsString("token","") );
    if(API_Token == ""){
        if(allow_example_firstRun)
            firstTimeRun_example();
        else
            printf("No Token Found - You need to do the first run code\n");
    }else{
        refreshSession();
    }
    //printf("Session token : %s\n",API_Session.toStdString().c_str());

    //testFunction();
    if(API_Session=="")
        return false;
    else
        return true;
}

void RedditSession::saveSettings(){
    GlopConfig::Settings set;
    set.values["token"] = API_Token.toStdString();
```

```
39      GlopConfig::SaveToFile((QDir::homePath()+"/.redditSettings.conf") .toStdString
40  }
41
42  void RedditSession::firstTimeRun_example(){
43      //this is an intereactive thing that runs when the settings don't seem to be
44      // https://github.com/reddit/reddit/wiki/OAuth2
45      std::string URL = firstTimeRun_authorizeURL("submit,read,history,edit,identity
46
47      printf("This is a first time setup\n");
48      printf("You need to visit the URL below and then authorize this program\n");
49      printf("%s\n",URL.c_str());
50      printf("\nAfter you authorize the program, it will send you to the reddit home
51      printf("\nCopy and paste the URL of the homepage to here\n");
52      std::getline(std::cin,URL);
53      printf("retrieving access token...\n");
54
55      firstTimeRun_redirectedURL(QString(URL.c_str()));
56  }
57  QString RedditSession::firstTimeRun_authorizeURL(QString wanted_scopes){
58      QString responseType = "code";
59      QString state = "should_be_random";
60      QString duration = "permanent";
61      //QString wanted_scopes = "submit,read,history,edit,identity,privatemessages
62
63      QString URL = QString("https://www.reddit.com/api/v1/authorize?")
64                    +"client_id="+clientID
65                    +"&response_type="+responseType
66                    +"&state="+state
67                    +"&redirect_uri="+redirectUri
68                    +"&duration="+duration
69                    +"&scope="+wanted_scopes;
70      return URL;
71
72  }
73  void RedditSession::firstTimeRun_redirectedURL(QString URL){
74      if(URL.indexOf("error")>=0 || URL.indexOf("code=")==-1){
75          printf("There was some sort of issue with the URL, try again\n");
76          exit(1);
77      }
78      QString code;
```

```
79      code = URL.right(URL.size() - URL.indexOf("code=") - QString("code=").size() )
80      code = code.left(code.indexOf("&")); // clean everything to the right of the
81      printf("one-time code : %s\n",code.toStdString().c_str());
82
83
84      QNetworkReply *reply =
85      makePOSTrequest("https://www.reddit.com/api/v1/access_token",
86                      QString(("grant_type=authorization_code&code="+code.toStdStrin
87                      );
88
89      //temp = QString::fromUtf8(reply->readAll());
90      //printf("reply : \n%s\n",temp.toStdString().c_str());
91      //QJsonDocument top = QJsonDocument::fromJson(temp.toUtf8());
92      QJsonDocument top = QJsonDocument::fromJson(reply->readAll());
93      if(top.isObject()){
94          QJsonObject obj = top.object();
95          API_Session = obj["access_token"].toString();
96          API_Token = obj["refresh_token"].toString();
97          saveSettings();
98      }else{
99          printf("Error: response from server was not valid json\n");
100         exit(0);
101     }
102 }
103
104 QNetworkReply* RedditSession::makePOSTrequest(QString URL, QString postData){
105     QNetworkRequest req;
106     commonRequestSetup(req,URL);
107     req.setHeader(QNetworkRequest::ContentTypeHeader,"application/x-www-form-urler
108     QNetworkReply *reply = networkManager->post(req,postData.toUtf8());
109     commonReplyWait(reply);
110     return reply;
111 }
112
113 QNetworkReply* RedditSession::makeGETrequest(QString URL){
114     QNetworkRequest req;
115     commonRequestSetup(req,URL);
116     QNetworkReply *reply = networkManager->get(req);
117     commonReplyWait(reply);
118     return reply;
```

```
}
void RedditSession::commonRequestSetup(QNetworkRequest &req,QString URL){
    if(networkManager == NULL){
        networkManager = new QNetworkAccessManager(this);
        connect( networkManager,SIGNAL(authenticationRequired(QNetworkReply*,QAuth
    }
    if(Session_Time_Used.elapsed() > 3570*1000) //in milliseconds -  30 seconds l
        refreshSession();
    if(RequestLimit_RequestsLeft <= 0)
        waitForRequestLimitReset();
    req.setUrl(QUrl(URL));
    req.setHeader(QNetworkRequest::UserAgentHeader,UserAgent);
    if(! (API_Session == ""))
        //req.setRawHeader("Authorization", ("bearer "+API_Session).toUtf8());
        req.setRawHeader("Authorization", "bearer "+API_Session.toLocal8Bit());
}
void RedditSession::commonReplyWait(QNetworkReply *reply){
    QEventLoop loop;
    connect(reply, SIGNAL(finished()), &loop, SLOT(quit()));
    loop.exec();

    if(reply->hasRawHeader("x-ratelimit-remaining")){
        RequestLimit_RequestsLeft = (int)reply->rawHeader("x-ratelimit-remaining")
        RequestLimit_TimeToReset = reply->rawHeader("x-ratelimit-reset").toInt();
        RequestLimit_TimeElapsed.restart();
        //printf("found header\n");
        //printf("left %s\n",reply->rawHeader("x-ratelimit-remaining").data());
        //printf("time %s\n",reply->rawHeader("x-ratelimit-reset").data());
        //printf("left %d time %d\n",RequestLimit_RequestsLeft,RequestLimit_Time
    }
}
void RedditSession::waitForRequestLimitReset(){
    printf("Exahsted requests for the time period, now waiting for %d seconds\n",R
    int delta = (RequestLimit_TimeToReset*1000) - RequestLimit_TimeElapsed.elapsed
    QEventLoop loop;
    QTimer timeLeft;
    connect(&timeLeft, SIGNAL(timeout()), &loop, SLOT(quit()));
    timeLeft.start(delta);
    loop.exec();
}
```

```
159
160 void RedditSession::sendAuthentication(QNetworkReply *reply, QAuthenticator *authe
161     if(! (API_Session == "")){ // should never get called - is hard set in the c
162         authenticator->setUser("bearer");
163         authenticator->setPassword(API_Session);
164         printf("authenticated with session token\n");
165     }else{ // only should happen on first API call while we are trying to get th
166         authenticator->setUser(clientID);
167         authenticator->setPassword("");
168         printf("authenticated with client ID\n");
169     }
170 }
171
172 void RedditSession::refreshSession(){
173     printf("Getting new session token\n");
174     QString req = "grant_type=refresh_token&refresh_token=" + API_Token;
175     QNetworkReply* reply = makePOSTrequest("https://www.reddit.com/api/v1/access_t
176
177     QJsonDocument top = QJsonDocument::fromJson(reply->readAll());
178     if(!top.isObject()){
179         printf("Error: response from server was not valid json, unable to refresh
180         exit(0);
181     }
182     QJsonObject obj = top.object();
183     API_Session = obj["access_token"].toString();
184
185     Session_Time_Used.restart();
186     //printf("\n%s\n\n",top.toJson().data());
187 }
188
189 void RedditSession::testFunction(){
190     //example json from the server when first authenticating the program to the
191     //{"access_token": "VnQBpiuRpH_otGfEqD30j7g_hZQ", "token_type": "bearer", "e
192 //    QByteArray input = "{\"access_token\": \"VnQBpiuRpH_otGfEqD30j7g_hZQ\", \"
193 //    QJsonDocument top = QJsonDocument::fromJson(input);
194 //    if(top.isObject());
195 //        printf("is object\n");
196 //    QJsonObject obj=top.object();
197 //    printf("Session %s\n",obj["access_token"].toString().toStdString().c_str()
198 //    printf("Refresh %s\n",obj["refresh_token"].toString().toStdString().c_str(
```

```
199
200     //------------------------------------------------------------------------
201     //a simple test for if the session token we get allows us to access things c
202     //auto reply = makeGETrequest("https://oauth.reddit.com/api/v1/me");
203     //QString temp = QString::fromUtf8(reply->readAll());
204     //printf("person: %s\n",temp.toLatin1().data());
205
206     //------------------------------------------------------------------------
207     // playing with reading data from a subreddit and what it returns
208     //auto reply = makeGETrequest("https://oauth.reddit.com/r/bottest");
209     //QString temp = QString::fromUtf8(reply->readAll());
210     //printf("subreddit data:\n%s\n",temp.toLatin1().data());
211
212     //printf("headers:\n");
213     //for(auto x : reply->rawHeaderPairs()){
214     //    printf("\t%s : %s\n",x.first.data(),x.second.data());
215     //}
216
217     //------------------------------------------------------------------------
218     //abuse test to test how well the "wait for request limit reset" functionali
219     //while(true){
220     //    makeGETrequest("https://oauth.reddit.com/r/bottest");
221     //    printf("%d reqs in %d seconds\n",RequestLimit_RequestsLeft,RequestLimi
222     //}
223 }
224
225 QString RedditSession::scopesDescriptions(){
226     return "\
227 creddits:         Spend my reddit gold creddits on giving gold to other users.\
228 modcontributors:  Add/remove users to approved submitter lists and ban/unban or mu
229 modmail:          Access and manage modmail via mod.reddit.com.\
230 modconfig:        Manage the configuration, sidebar, and CSS of subreddits I moder
231 subscribe:        Manage my subreddit subscriptions. Manage 'friends' - users whos
232 structuredstyles: Edit structured styles for a subreddit I moderate.\
233 vote:             Submit and change my votes on comments and submissions.\
234 wikiedit:         Edit wiki pages on my behalf\
235 mysubreddits:     Access the list of subreddits I moderate, contribute to, and sub
236 submit:           Submit links and comments from my account.\
237 modlog:           Access the moderation log in subreddits I moderate.\
238 modposts:         Approve, remove, mark nsfw, and distinguish content in subreddit
```

```
modflair:          Manage and assign flair in subreddits I moderate.\
save:              Save and unsave comments and submissions.\
modothers:         Invite or remove other moderators from subreddits I moderate.\
read:              Access posts and comments through my account.\
privatemessages:   Access my inbox and send private messages to other users.\
report:            Report content for rules violations. Hide & show individual subm
identity:          Access my reddit username and signup date.\
livemanage:        Manage settings and contributors of live threads I contribute to
account:           Update preferences and related account information. Will not hav
modtraffic:        Access traffic stats in subreddits I moderate.\
wikiread:          Read wiki pages through my account\
edit:              Edit and delete my comments and submissions.\
modwiki:           Change editors and visibility of wiki pages in subreddits I mode
modself:           Accept invitations to moderate a subreddit. Remove myself as a m
history:           Access my voting history and comments or submissions I've saved
flair:             Select my subreddit flair. Change link flair on my submissions.'
}
```

C main.cpp

```
#include <QCoreApplication>
#include "redditapi.h"

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    RedditSession *ra = new RedditSession;
    QMetaObject::invokeMethod(ra,"parseSettings",Qt::QueuedConnection);
    //QMetaObject::invokeMethod(ra,"testFunction",Qt::QueuedConnection);

    return a.exec();
}
```